Jahan Ulhaque
ID: 201272455

COMP219 - A1.1

SIMPLE MACHINE LEARNING MODEL

Checklist: All complete: 7/7 And Extra

- ✓ Step 1: Loading Data
- ✓ Step 2: Training
- ✓ Step 2.1: Code for training
- ✓ Step 2.2: Successful training
- ✓ Step 3: Model Evaluation
- ✓ Step 3.1: Explain your experimental design
- ✓ Step 3.2 Document your evaluation results
- ✓ Extra 20%

HOW TO RUN:

Please read the "README-How_to_run" file contained within the folder.

STEP 3: MODEL EVALUATION

I have used a "banknote authentication" data sample. The data was extracted from genuine and fake banknotes. Using wavelet transform tool they could extract features from the images.
The data structure is in all integer format

Link: https://archive.ics.uci.edu/ml/datasets/banknote+authentication#

Attribute information:

1. variance of Wavelet Transformed image (continuous)
2. skewness of Wavelet Transformed image (continuous)
3. curtosis of Wavelet Transformed image (continuous)
4. entropy of image (continuous)
5. class (integer)

```
Variance,Skewness,Curtosis,Entropy,Class
3.6216,8.6661,-2.8073,-0.44699,0
4.5459,8.1674,-2.4586,-1.4621,0
3.866,-2.6383,1.9242,0.10645,0
3.4566,9.5228,-4.0112,-3.5944,0
0.32924,-4.4552,4.5718,-0.9888,0
4.3684,9.6718,-3.9606,-3.1625,0
3.5912,3.0129,0.72888,0.56421,0
2.0922,-6.81,8.4636,-0.60216,0
3.2032,5.7588,-0.75345,-0.61251,0
1.5356,9.1772,-2.2718,-0.73535,0
1.2247,8.7779,-2.2135,-0.80647,0
3.9899,-2.7066,2.3946,0.86291,0
1.8993,7.6625,0.15394,-3.1108,0
-1.5768,10.843,2.5462,-2.9362,0
3.404,8.7261,-2.9915,-0.57242,0
4.6765,-3.3895,3.4896,1.4771,0
2.6719,3.0646,0.37158,0.58619,0
0.80355,2.8473,4.3439,0.6017,0
1.4479,-4.8794,8.3428,-2.1086,0
5.2423,11.0272,-4.353,-4.1013,0
5.7867,7.8902,-2.6196,-0.48708,0
0.3292,-4.4552,4.5718,-0.9888,0
3.9362,10.1622,-3.8235,-4.0172,0
0.93584,8.8855,-1.6831,-1.6599,0
4.4338,9.887,-4.6795,-3.7483,0
0.7057,-5.4981,8.3368,-2.8715,0
1.1432,-3.7413,5.5777,-0.63578,0
-0.38214,8.3909,2.1624,-3.7405,0
```

Summary:

I have implemented both decision tree and KNN (k nearest neighbour) algorithm. Furthermore I'd like to add, I have extended the KNN algorithm by allowing the program to find the best KNN value, which I will explain later.

STEP 3.1: EXPLAIN YOUR EXPERIMENTAL DESIGN:

I have used the following methods to evaluate my trained model.

- Accuracy score, Confusion matrix, Cross validation and Classification report
- Accuracy score:

The "accuracy score" functions works by doing the following:
It will loop through the predicted sample and the expected true sample, and for every correct answer (those which match correctly) it will store this value and then divide by the number of samples.
i.e., given a sample size of 275, and we predict 272 instances correctly, then it will do 275/272, thus it will give the following:

```
Accuracy on test sample is: 0.9890909090909091/1
```

- Confusion matrix

Confusion matrix allows us to work out how many instances we predicted correctly and incorrectly. So, given a 2 by 2 matrix:

```
Confusion Matrix:
[[146   8]
 [  2 119]]
```

The bottom right value, and top left value shows us the predictions we got correct.
The bottom left value, and top right value shows us the predictions we got incorrectly.
By totalling up the numbers from the confusion matrix, we get the total test instances. Given the high accuracy of this classifier and shown from the confusion matrix the amount of incorrectly predicted values is very low.

- Cross validation

In k-fold cross-validation, the original sample is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once. For example:

reference:
https://www.openml.org/a/estimation-procedures/1

```
10 Fold cross validation:
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

10 Fold cross validation average/mean:
 1.0
```

- Classification report

The classification report is a further add on to the confusion matrix, basically explaining what the confusion matrix means.

It computes precision, recall, F-measure and support for each class. In our case, we have two classes, 0 and 1.

```
Classification report:
              precision    recall  f1-score   support

          0       0.99      0.99      0.99       149
          1       0.98      0.99      0.99       126

avg / total       0.99      0.99      0.99       275
```

Precision: The precision is used to classifier the data and to assign the correct labels. i.e. not to label positive sample that is actually negative.

Recall: The recall is intuitively the ability of the classifier to find all the positive samples.

f1-score: The F1-score can be interpreted as a weighted harmonic mean of the precision and recall, where an F1-score reaches its best value at 1 and worst score at 0.

Support: The support is the number of samples of the true response that lie in that class.

STEP 3.2: DOCUMENT YOUR EVALUATION RESULTS:

Decision tree algorithm: (real output from the code)

```
Decision Tree:
Amount of Entities: 1372,
Amount of classes per entity: 5
Test sample size and classes: (275, 4)

Model evaluation:

Confusion Matrix:
 [[151   6]
 [  4 114]]

Accuracy on test sample is: 0.9636363636363636/1

Classification report:
              precision    recall  f1-score   support

          0       0.97      0.96      0.97       157
          1       0.95      0.97      0.96       118

avg / total       0.96      0.96      0.96       275


10 Fold cross validation:
 [0.98550725 0.99275362 0.96350365 0.99270073 0.97080292
0.98540146
 0.99270073 0.99270073 0.99270073 0.97810219]

10 Fold cross validation average/mean:
 0.984687400825135
```

By doing the traditional means of splitting the data into training and test (70% training, 30% testing in my case), we have computed the following results (as shown above).
The confusion matrix and accuracy were done on the test sample, which was out of 275 instances, containing 4 labels per instance. We can see from the results that this classifier is very accurate (achieving 0.963% / 1).

Now there is a chance overfitting can occur, due to the algorithm fitting the data too well. Or underfitting occurring, which is when the algorithm cannot capture the underlying trend from the data and thus does not fit the data very well. This occurs if the algorithm shows high bias and low variance.

Now to solve this possible issue, we can use cross validation, which gives us more true accurate results, but runs K times slower than the train/split method.

By using cross validation, we reduce errors caused by bias, and every data point gets to be in the validation set exactly one time. this reduces bias, as we are using most of the data for testing and reduces variance as most of the data is being used in the validation set.

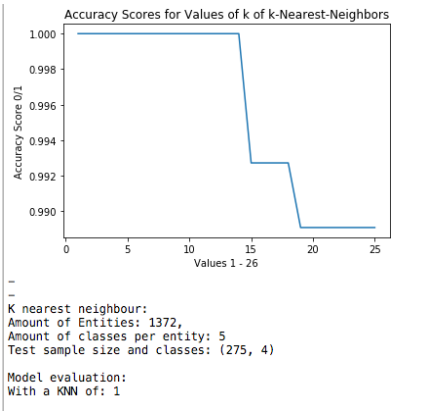
So now, using 10-fold cross validation, we can get more accurate data. As we can see from the results, all 10 cross validation accuracy is higher or sometimes on par with the train/test split method. Thus, the average mean is also higher, at 0.984%/1 compare to 0.963%/1. This means cross validation gives you a more accurate score.

The same rule can be applied to my next attempt at applying another algorithm to this data set.

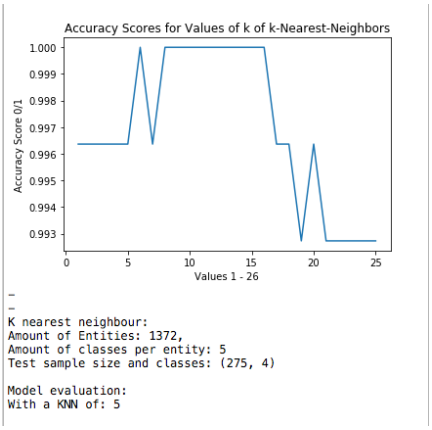
K nearest neighbour algorithm: (real output from the code)

```
K nearest neighbour:
Amount of Entities: 1372,
Amount of classes per entity: 5
Test sample size and classes: (275, 4)

Model evaluation:
With a KNN of: 3

Confusion Matrix:
[[149   0]
 [  0 126]]

Accuracy on test sample is: 1.0/1

Classification report:
              precision    recall  f1-score   support

          0       1.00      0.99      1.00       149
          1       0.99      1.00      1.00       126

avg / total       1.00      1.00      1.00       275


10 Fold cross validation:
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

10 Fold cross validation average/mean:
 1.0
```

Within my code, I loop through 1 - 26 and assign K to each of the iterations, I then store the accuracy score of each iteration. I then pick assign the KNN value to the index which gave the best accuracy score. I have noticed KNN is usually 1, 3 or 5. I have displayed a graph for my testing to see which KNN value is ideal and to see the ranges of accuracy scores. For example:




```
K nearest neighbour:
Amount of Entities: 1372,
Amount of classes per entity: 5
Test sample size and classes: (275, 4)

Model evaluation:
With a KNN of: 1
```




```
K nearest neighbour:
Amount of Entities: 1372,
Amount of classes per entity: 5
Test sample size and classes: (275, 4)

Model evaluation:
With a KNN of: 5
```

Without cross validation, I get usually get an accuracy score of 0.99 or 1. And using cross validation is no different, the accuracy is still very high achieving 1/1 accuracy 9/10 times. We also see that the confusion matrix and classification report is higher than the decision trees results.

COMPARING RESULTS AND SUMMARY:

As we can see, KNN is better than decision tree in both testing instances, either via training/testing split or cross validation, gaining a higher accuracy score every time. Out of the two algorithms KNN is the better option. I think this is because when you're dealing with just numbers its more difficult to use a decision tree classifier as any small change can result in a completely different tree. compare to KNN, which will just use the KNN values to predict the correct answer, and we were able to further improve this by selecting the best KNN value.